Tema 1 IA

Nicolescu Radu-Catalin, 343C4

1. Detalii de implementare

Pentru rezolvarea cerintei 0, am implementat functiile init\_env, care citeste fisierul de input si populeaza structurile de date si variabilele necesare functionarii algoritmilor si get\_next\_states, care va crea un dictionar care mapeaza pentru fiecare nod, nodurile in care se poate deplasa agentul.

Nu am avut nevoie de implementarea functiei apply\_action, deoarece in contextul temei, aplicarea unei actiuni inseamna deplasarea dintr-un nod intr-un nod adiacent valabil, iar aceasta actiune se poate face modificand direct nodul curent cu un nod vecin.

De asemenea, functia get\_next\_states intoarce de la inceput un dictionar cu nodurile vecine valabile ale fiecarui nod din graf.

Pe langa variabilele sursa, destinatie, am avut nevoie de urmatoarele structuri de date:

* pos\_dict -> face legatura intre id-ul nodului si coordonatele sale + daca acesta este obstacol
* adj\_list -> face legatura intre id-ul nodului curent si id-urile vecinilor(fie ca sunt obstacole sau nu)
* cost\_dict -> retine costurile muchiilor intre 2 id-uri de noduri
* next\_sts -> face legatura intre id-ul nodului curent si id-urile vecinilor care nu sunt obstacole
* prev\_a, prev\_s, result, H -> folosite in implementarea LRTA\*

Implementarile DFID, IDA\* si LRTA\* au fost realizate dupa pseudocodul oferit in cursurile “Strategii de cautare” si “Cautari online”, cu doua imbunatatiri pentru primii 2 algoritmi mai sus mentionati:

1. am precalculat vecinii in dictionarul next\_sts, pentru a evita calcularea inutila a acestora la fiecare apel de functie (harta ramane neschimbata pe parcursul cautarilor, deci nu are sens repetarea calcularii)

2. am limitat explorarea inutila a nodurilor prin crearea unui dictionar care retine cel mai bun cost pentru un nod -> se exploreaza acel nod doar daca noul cost calculat pentru acesta este strict mai mic decat costul din dictionar

Mod de rulare: python3 tema.py

<introducere numar fisier de input (1,2,3)>

1. Alegerea euristicii

Euristica aleasa de mine a fost , care este mai mica decat distanta Manhattan pe baza inegalitatii mediilor:

unde x2, y2 sunt coordonatele destinatiei.

Distanta Manhattan subestimeaza costul real pentru a ajunge din sursa in destinatie in tema datorita prezentei obstacolelor si a pozitiilor invalide pe harta.

La randul ei, euristica mea subestimeaza distanta Manhattan datorita inegalitatii mediilor.

Astfel,

unde:

h(s) – euristica mea

Manhattan(s) – distanta Manhattan

h\*(s) – costul real

Avand in vedere cele de mai sus, euristica mea este admisibila.

1. Rezultate obtinute si analiza acestora.

Algoritmii implementati obtin aceleasi costuri:

* Input1 -> cost 18
* Input2 -> cost 89
* Input3 -> cost 139

Acestia vor fi analizati din punct de vedere al mediei timpilor obtinuti in urma a 5 rulari, pe input1, input2, input3.

Rezultatele obtinute pe input1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | T1  (ms) | T2  (ms) | T3  (ms) | T4  (ms) | T5  (ms) | Medie  (ms) |
| DFID | 5,083 | 6,152 | 5,786 | 5,840 | 5,527 | 5,678 |
| IDA\*  (Euclidian) | 5,209 | 4,684 | 4,516 | 4,701 | 4,173 | 4,657 |
| IDA\*  (euristica mea) | 3,021 | 3,142 | 2,846 | 3,047 | 2,788 | 2,969 |
| LRTA\*  (Euclidian) | 4,752 | 5,637 | 4,998 | 4,693 | 5,064 | 5,029 |
| LRTA\*  (euristica mea) | 5,417 | 5,322 | 5,358 | 4,985 | 5,374 | 5,291 |

Rezultatele obtinute pe input2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | T1  (ms) | T2  (ms) | T3  (ms) | T4  (ms) | T5  (ms) | Medie  (ms) |
| DFID | 64,844 | 66,625 | 65,937 | 64,229 | 64,833 | 65,294 |
| IDA\*  (Euclidian) | 134,474 | 136,179 | 136,187 | 138,103 | 140,074 | 137,003 |
| IDA\*  (euristica mea) | 102,949 | 103,339 | 104,610 | 103,137 | 102,170 | 103,241 |
| LRTA\*  (Euclidian) | 29,129 | 29,555 | 28,688 | 28,736 | 29,309 | 29,083 |
| LRTA\*  (euristica mea) | 28,138 | 28,999 | 27,929 | 27,728 | 28,067 | 28,172 |

Rezultatele obtinute pe input3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | T1  (ms) | T2  (ms) | T3  (ms) | T4  (ms) | T5  (ms) | Medie  (ms) |
| DFID | 436,110 | 512,313 | 441,463 | 486,242 | 483,133 | 471,852 |
| IDA\*  (Euclidian) | 975,342 | 1017,423 | 1021,749 | 985,193 | 988,601 | 997,662 |
| IDA\*  (euristica mea) | 638,704 | 673,605 | 651,453 | 652,745 | 654,347 | 654,171 |
| LRTA\*  (Euclidian) | 121,630 | 123,772 | 123,521 | 120,027 | 120,282 | 121,846 |
| LRTA\*  (euristica mea) | 108,005 | 121,356 | 111,025 | 109,107 | 119,661 | 113,831 |

Asa cum era de asteptat, algoritmii din familia DFID (DFID si IDA\*) sunt mai lenti fata de LRTA\* datorita modificarii dinamice a costului maxim admisibil si a reluarii cautarii din sursa, cu costul maxim admisibil actualizat de la o executie a buclei din DFID\_loop sau IDA\_loop la alta.

Pe de alta parte, LRTA\* gaseste in timp constant cate o noua pozitie, gasind dupa prima rulare o cale neoptima, lunga, cu cicluri, insa pe baza careia isi actualizeaza tabela de evaluari euristice, ajungand in final sa convearga catre o cale de cost optim.

Se observa o diferenta sesizabila de viteza in favoarea euristicii implementate de mine in cazul IDA\*, insa in cazul LRTA\*, diferenta este insesizabila, cele doua grafice aproape suprapunandu-se.

In ceea ce priveste euristicile, nu exista o ordonare stricta a acestora pentru a putea spune ca o euristica este dominata de cealalta si sa concluzionam ca un algoritm care o foloseste pe una este mai informat decat cel care o foloseste pe cealalta.

<>

unde a = |x1-x2| si b = |y1-y2|